OIPE
NOV 1 7 2005
IAP87
PATENT & TRADEMARK OFFICE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

NCR Docket No. 10157

In re Application of:

Rolf Günter Erich Stegelmann et al. Group Art Unit: 2167

Application No.: 10/081,079 Examiner: Wassum, Luke S.

Filed: 02/21/2002

For: EVALUATING EXPRESSIONS IN STORED PROCEEDURES

COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

Best Available Copy

DECLARATION OF FACTS UNDER 37 C.F.R. §1.131

I, Jeetendra Chaware, hereby declare under penalty of perjury under the laws of the United States of America that:

1. I am a co-inventor of the invention described and claimed in U.S. Patent Application 10/081,079, filed on February 21, 2002 entitled "Evaluating Expressions in Stored Procedures;"

2. The invention described and claimed in the above application was reduced to practice on or before January 4, 2001 as evidenced by the proprietary document entitled "Performance Characterization For Teradata RDBMS Stored Procedures (TDSP) Feature In V2R4.1," included herewith in a sealed envelope (*see, e.g.*, pg. 1, lines 4-5, 21-24 & 34-35; pg. 2, lines 1-19 & 38-47; pg. 5, lines 16-23);

3. All statements made of my knowledge are true and all statements made on information and belief are believed to be true; and

4. I understand that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and may jeopardize the validity of this application or any patent issuing thereon.

[illegible]

Jeetendra Chaware

Date: 03-NOV-2005

OIPE IAP87
NOV 17 2005
PATENT & TRADEMARK OFFICE

NCR

Performance Characterization

For

Teradata RDBMS Stored Procedures (TDSP) Feature In V2R4.1

January 4, 2001

Author
Varshali Patwari

Contributors
Manjula Koppuravuri, Rethi Ravindran, Jeetendra Chaware

NDF@Satyam, Hyderabad

Teradata Development Division
Teradata Solutions Group

THIS DOCUMENT AND INFORMATION HEREIN ARE THE PROPERTY OF NCR CORPORATION AND ALL UNAUTHORIZED USE AND REPRODUCTION ARE PROHIBITED

Copyright © 1999-2001 by NCR Corporation, Dayton, Ohio U.S.A.
All Rights Reserved. Printed in U.S.A.
CONFIDENTIAL
Unpublished Property of NCR Corporation

TABLE OF CONTENTS

TERADATA STORED PROCEDURES PERFORMANCE CHARACTERIZATION 1

PURPOSE 1
PERFORMANCE TEST ENVIRONMENT 1
Test Scenarios *2*
Stored procedure with multiple SPL statements and a single SQL statement 2
Stored procedure with multiple SPL statements and multiple SQL statements 4
OTHER PERFORMANCE CRITERIA 7
Performance Test Environment *8*
Test Stored Procedures *8*
Resource Usage *9*
Collection of Performance Data using Resource Usage Macros 9
Collection of Performance data using System Activity Reporter (SAR) 12
OVERALL CONCLUSIONS *12*
RESOURCE USAGE TEST SCRIPTS 14
SYSTEM CONFIGURATIONS 19
SMP 4400 (pennar) *19*
MPP 5250 (agni) *20*

Teradata Stored Procedures Performance Characterization

Purpose

This report contains the performance characterization for the Stored Procedures feature in Teradata V2R4.1 MP-RAS release. This is part of the performance improvements targeted for stored procedures feature in V2R4.1 release. Refer to the *TDSP Requirements Specification* 541-0002297A and *TDSP DFES* 541-0002210A for the performance improvement requirement.

Performance Test Environment

The following systems were used for performance testing: (refer to System Configurations on page 19 for details on the configurations)

- an SMP system – 4400
- an MPP (4 nodes) system – 5250

The performance testing was carried out on the systems ***with load, without load, with object code caching*** and ***without object code caching feature***.

- **without load:** ensured that the system is in quiescent state while starting the performance testing and no other user is using the DBS during the testing.
- **with load:** ensured that the system is loaded throughout the testing period by executing an SQL CALL statement repeatedly through four different sessions. This was done to ensure constant workload on the components like Parser and RTS.

Performance testing with and without load is required because in V2R4.0 the expressions specified inside SPL source text are evaluated as an SQL SELECT statement. In contrast, in V2R4.1, an EVL code is generated for the expressions. Hence, the load/no load condition helps in measuring the performance gain in V2R4.1 as no SQL processing is required for expression evaluation.

- **with object code caching feature:** ensured that the same stored procedure is executed repeatedly for different number of iterations. As the stored procedure names are same, the object code caching feature is also enabled.
- **without object code caching feature:** ensured that stored procedures with the same definition but different procedure names are executed. As the stored procedure names are different, the benefits of object code caching feature are not utilized. Each stored procedure is executed with a different value for iteration.

Performance testing with and without object code caching is required because, in V2R4.0, for every CALL statement, the object code of the stored procedure is fetched from the stored procedure table and the stored procedure is executed. In contrast, in V2R4.1, the object code corresponding to a stored procedure is fetched and cached during the first execution of the stored procedure. For any subsequent execution of the same stored procedure within a time period of 10 minutes the object code is picked up from the cache. This results in performance gain in terms of reduced number of fetches from the stored procedure table.

The test cases have been chosen in such a way that the SQL statements within the stored procedures are executed only once. This is to ensure that the execution time of the SQL statements does not impact the overall performance. (The performance improvement in V2R4.1 was targeted for expression evaluation and object code caching and **not** for SQL statement execution). This is because the execution time of SQL statements is more compared to the execution time of EVL-code of an SPL SET statement or an expression.

The following table provides the recorded impact on performance due to execution of an SQL statement:

S.No.	Test Scenario	Time on V2R4.0	Time on V2R4.1	Improvement
1.	Stored procedure containing an SQL INSERT statement and an SPL SET statement within a WHILE loop. The WHILE loop is executed 3000 times.	[redacted]	[redacted]	[redacted]
2.	Stored procedure containing only an SPL SET statement within a WHILE loop. The WHILE loop is executed 3000 times.	[redacted]	[redacted]	[redacted]
3.	Stored procedure containing 3000 SQL INSERT statements and **no** other expression.	[redacted]	[redacted]	[redacted]

From the above statistics, it can be inferred that there is a performance improvement if the stored procedures contains expressions. Almost no performance improvement is gained for the stored procedures executing only SQL statements.

Also, it can be observed that execution of a stored procedure containing 3000 SQL INSERT statements takes significantly more time when compared to the execution of a stored procedure containing a SPL SET statement and a WHILE condition expression 3000 times. It can be observed that as the number of SQL statements increase in a stored procedure along with expressions, overall performance gain is less in V2R4.1 as compared to V2R4.0.

Test Scenarios

Stored procedure with multiple SPL statements and a single SQL statement

The following is a stored procedure containing an SQL INSERT statement and an SPL SET statement in a WHILE loop. The SQL INSERT statement at line # 6 gets executed only once when the condition at line #5 gets evaluated to true.

```
1.        REPLACE PROCEDURE pe(IN i INTEGER, INOUT k INTEGER)
2.        BEGIN
3.            WHILE (k <= i)
4.            DO
5.                IF (k = i) THEN
6.                     INSERT load1(:i);
7.                END IF;
8.                SET k = k + 1;
9.            END WHILE;
10.       END;
```

In V2R4.0, for each execution of the above stored procedure, three SQL SELECT statements are executed, one for each of the following:

- the conditional expression at line #3,
- the conditional expression at line #5, and
- the SPL SET statement at line #8

Hence, in V2R4.0, totally (3 * i + 1) SQL statements are executed.

In contrast, in V2R4.1, no SQL statement is executed for the SPL expressions. Those are evaluated locally within the stored procedure using the EVL code generated for the expressions during stored procedure creation. Hence, in the entire stored procedure execution, the SQL INSERT statement at line #6 is executed only once resulting in a performance gain. The results are as follows:

Caching enabled

On SMP – With load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Smp-Load-Cache.xls):

Smp-Load-Cache.xls

On SMP – With no load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Smp-NoLoad-Cache.xls):

Smp-NoLoad-Cache.
xls

On MPP – With load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Mpp-Load-Cache.xls):

Mpp-Load-Cache.xls

On MPP – With no load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Mpp-NoLoad-Cache.xls):

Mpp-NoLoad-Cache.
xls

Caching not enabled

For disabling the object code caching feature, multiple stored procedures such as the following, are created with the same definition, but with different names, e.g., pe1, pe2 etc. Different values are passed for the parameter that controls the number of iterations of the WHILE loop.

```
REPLACE PROCEDURE pe1(IN i INTEGER, INOUT k INTEGER)
BEGIN
    WHILE (k <= i)
    DO
        IF (k = i) THEN
             INSERT load1(:i);
        END IF;
        SET k = k + 1;
    END WHILE;
END;
```

```
REPLACE PROCEDURE pe2(IN i INTEGER, INOUT k INTEGER)
BEGIN
    WHILE (k <= i)
    DO
        IF (k = i) THEN
```

```
                    INSERT load1(:i);
               END IF;
               SET k = k + 1;
          END WHILE;
     END;
```

The results are as follows:

On SMP – with load – caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Smp-Load-NoCache.xls):

Smp-Load-NoCache.
xls

On SMP – With NO load – Caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Smp-NoLoad-NoCache.xls):

Smp-NoLoad-NoCach
e.xls

On MPP – With load – Caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Mpp-Load-NoCache.xls):

Mpp-Load-NoCache.
xls

On MPP – With NO load – Caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (Mpp-NoLoad-NoCache.xls):

Mpp-NoLoad-NoCach
e.xls

Conclusion

The stored procedure execution time is less in V2R4.1 as compared to V2R4.0. The improvement ranges from [redacted]% to [redacted]%.

Stored procedure with multiple SPL statements and multiple SQL statements

The stored procedure definition is as follows:

```
REPLACE PROCEDURE PEMulti (IN i INTEGER, INOUT k INTEGER)
BEGIN
     WHILE (k <= i)
      DO
```

```
5.                      IF (k = i) THEN
6.                           INSERT load1(:k);
7.                      ELSEIF (k = i - 1) THEN
8.                           DELETE load1;
9.                      ELSEIF (k = i - 2) THEN
10.                          INSERT load1(:k);
11.                     ELSEIF (k = i - 3) THEN
12.                          DELETE load1;
13.                     ELSEIF (k = i - 4) THEN
14.                          INSERT load1(:k);
15.                     END IF;
16.                     SET k = k + 1;
17.               END WHILE;
18.          END;
```

In V2R4.0, for each execution of the above stored procedure, seven SQL SELECT statements are executed, one for each of the following:

- the conditional expressions at lines #4, #6, #8, #10, #12, and #14
- the SPL SET statement at line #17

Hence, in V2R4.0, totally (7 * i + 1) SQL statements are executed.

In contrast, in V2R4.1, no SQL statement is executed for the SPL expressions. Those are evaluated locally within the stored procedure using the EVL code generated for these expressions during stored procedure creation. The results are as follows:

Caching enabled

On SMP – With load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (SmpMultiSQL-Load-Cache.xls):

SmpMultiSQL-Load-C
ache.xls

On SMP – With NO load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (SmpMultiSQL-NoLoad-Cache.xls):

SmpMultiSQL-NoLoac
-Cache.xls

On MPP – With load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (MppMultiSQL-Load-Cache.xls):

MppMultiSQL-Load-C
ache.xls

On MPP – With NO load – Caching enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (MppMultiSQL-NoLoad-Cache.xls):

MppMultiSQL-NoLoad
-Cache.xls

Caching not enabled

For disabling the object code caching feature, multiple stored procedures such as the following, are created with the same definition, but with different names, e.g., PEMulti1, PEMulti2, etc. Different values are passed for the parameter that controls the number of iterations of the WHILE loop.

```
REPLACE PROCEDURE PEMulti1 (IN i INTEGER, INOUT k INTEGER)
BEGIN
     WHILE (k <= i)
      DO
            IF (k = i) THEN
                 INSERT load1(:k);
            ELSEIF (k = i - 1) THEN
                 DELETE load1;
            ELSEIF (k = i - 2) THEN
                 INSERT load1(:k);
            ELSEIF (k = i - 3) THEN
                 DELETE load1;
            ELSEIF (k = i - 4) THEN
                 INSERT load1(:k);
            END IF;
            SET k = k + 1;
     END WHILE;
END;
```

```
REPLACE PROCEDURE PEMulti2 (IN i INTEGER, INOUT k INTEGER)
BEGIN
     WHILE (k <= i)
      DO
            IF (k = i) THEN
                 INSERT load1(:k);
            ELSEIF (k = i - 1) THEN
                 DELETE load1;
            ELSEIF (k = i - 2) THEN
                 INSERT load1(:k);
            ELSEIF (k = i - 3) THEN
                 DELETE load1;
            ELSEIF (k = i - 4) THEN
                 INSERT load1(:k);
            END IF;
            SET k = k + 1;
     END WHILE;
END;
```

The results are as follows:

On SMP – With load – Caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (SmpMultiSQL-Load-NoCache.xls):

SmpMultiSQL-Load-N
oCache.xls

On SMP – With NO load – Caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (SmpMultiSQL-NoLoad-NoCache.xls):

SmpMultiSQL-NoLoac
-NoCache.xls

On MPP – With load – Caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (MppMultiSQL-Load-NoCache.xls):

MppMultiSQL-Load-N
oCache.xls

On MPP – With NO load – Caching NOT enabled: Performance improvement is [redacted]% to [redacted]%. See details in the spreadsheet attached below (MppMultiSQL-NoLoad-NoCache.xls):

MppMultiSQL-NoLoad
-NoCache.xls

Conclusion

The stored procedure execution time is less in V2R4.1 as compared to V2R4.0. The overall performance improvement ranges from [redacted]% to [redacted]%. In addition, as the number of SQL statements increase in a stored procedure along with an expression, the overall performance gain is less in V2R4.1 as compared to V2R4.0.

Also, the performance gain is higher when caching is enabled (ranged from [redacted]% to [redacted]%).

Other Performance Criteria

The performance report so far contains the performance characterization of stored procedures in V2R4.1 release with respect to the elapsed time of stored procedure execution.

The performance enhancements in stored procedures V2R4.1 release also have an impact on the resource usage – CPU usage, I/O cost usage during stored procedure execution in the following manner:

- **CPU Usage:** In V2R4.0, SQL SELECT statements were executed to evaluate the SPL expressions. Due to this, CPU usage was more in Parser and Dispatcher partitions of PE and less in the partition used for RTS. In V2R4.1, as the expressions are evaluated in RTS, the CPU usage is less in Parser and Dispatcher partitions and more in the partition of PE used for RTS. The ResUsage tables do not provide the details of the RTS partition, and the data has been used from the 'Misc' partition data.

- **I/O Cost:** The execution of stored procedures in V2R4.1 impacts the I/O cost related to AMP disks as well as Unix file system in the following manner:

  - Impact on I/O cost on AMP disks

  In V2R4.0 stored procedures, an express request is submitted for fetching the object code of a stored procedure during each execution of the stored procedure. With the introduction of object-code caching feature in V2R4.1, express requests are sent to fetch the object code only for the first execution of stored procedure. The fetched object code is available in the cache for execution till the end of a timeout period. This reduces the number of express requests in the execution of stored procedures on V2R4.1 as compared to V2R4.0, for example, in nested procedure scenarios and when a stored procedure is executed repeatedly.

- Impact on I/O in Unix File System

The object code that is fetched during stored procedure execution is temporarily saved in Unix File System as a separate file. With the introduction of object-code caching feature in V2R4.1, this temporary file is created only for the first execution of the stored procedure, as against V2R4.0 where it is created for every execution of the stored procedure. This reduces the I/O on Unix File System in V2R4.1 stored procedure execution as compared to V2R4.0.

Performance Test Environment

As the CALL request for a stored procedure execution is always directed to one node (the node containing the AMP where the object code is stored), the performance impact in resource usage is the same on a SMP as well as a MPP. Hence, the resource usage data for the performance testing was collected on a SMP (4400) system.

Test Stored Procedures

A nested stored procedure test case was used for the following reasons.

- Object code fetching - I/0 cost
- Expressions - CPU usage

Outer Stored Procedure

```
REPLACE PROCEDURE penestl(IN i INTEGER)
BEGIN
        DECLARE k INTEGER DEFAULT 1;
        WHILE (k <= i)
         DO
              CALL penestl1();
              SET k = k + 1;
         END WHILE;
END;
```

Inner (nested) Stored Procedure

The inner stored procedure has 100 SPL IF statements, each containing an SPL SET statement within it. The entire stored procedure definition is not shown here. This test case was chosen as larger the SPL text larger is the object code for it, due to the presence of the EVL code. The interpretation of data corresponding to I/O costs is less error-prone, when the sizes are considerable.

```
REPLACE PROCEDURE penestl1()
BEGIN
     DECLARE p INTEGER;
     IF (1 = 1) THEN
         SET p = 1;
     END IF;
     IF (1 = 1) THEN
         SET p = 2;
     END IF;
          ...
          ...
```

```
            ...
          IF (1 = 1) THEN
              SET p = 100;
          END IF;
      END;
```

Resource Usage

The resource sampling is enabled on the database and Resource Usage Macros are used for collecting the performance data for CPU Usage and I/O cost. In addition, System Activity Reporter (SAR) of Unix was used to test the performance gain in I/O cost related to Unix File System during stored procedure execution.

Collection of Performance Data using Resource Usage Macros

Customized resource usage macros were developed for this purpose, to facilitate capturing of data related to only CPU and I/O usage. The Resource Usage Test Scripts section describes the details of customized resource usage macros.

Note:

- The ResUsage table (DBC.ResUsageSVPr) provides the statistics on a partition basis. RTS is in the Console partition of the PE Vproc. Currently, Parser/Dispatcher/Session Control partition related data is available, and the data corresponding to the activity of any other partitions is being logged as 'Misc' data. The RTS data is analyzed using the 'Misc' data.
- User service is the time that a CPU is busy executing user service code, which is privileged work performing system-level services on behalf of user execution processes.
- User execution is the time a CPU is busy executing user execution code, which is the time spent in a user state on behalf of a process.

The test script used is available in Resource Usage Test Scripts.

Collection rate Used: 10 Seconds. The statistics are collected once in every collection rate period.

Logging rate Used: 10 Seconds. The statistics are logged once in every collection rate period.

Three nested stored procedures test cases having the same definition as mentioned in Test Stored Procedures, were used to collect the performance data for 1000, 2000 and 3000 iterations so that the minimum stored procedure execution time is between 40-70 seconds. The total execution time spans between 4-7 logging time intervals and helps in enhancing the data accuracy.

The following two customized resource usage macros are used:

1. ResCpuByPEOneNode
2. ResOneNode

ResCPUByPEOneNode: This gives the CPU busy percentage for the following:

- User Service for the Parser partition of the PE. See details in the spreadsheet attached below (CPUTimeForServiceForParserPartition.xls):

CPUTimeForServiceF
orParserPartition.xls

Conclusion

The percentage CPU usage for service for the Parser partition of PE is less in V2R4.1 as compared to V2R4.0 (improvement ranges from [redacted]% to [redacted]%). The reason is that in V2R4.1 no SQL statements are submitted to parser for executing the expressions.

- User Service for the Dispatcher partition of the PE. See details in the spreadsheet attached below (CPUTimeForServiceForDispatcherPartition.xls):

CPUTimeForServiceF
orDispatcherPartition

Conclusion

The percentage CPU usage for service for the Dispatcher partition of PE is less in V2R4.1 as compared to V2R4.0 (improvement ranges from [redacted]% to [redacted]%).

- User Service for the Miscellaneous partition of the PE. See details in the spreadsheet attached below (CPUTimeForServiceForMiscPartition.xls):

CPUTimeForServiceF
orMiscPartition.xls

Conclusion

The percentage CPU usage for service for Miscellaneous partitions of PE is more in V2R4.1 as compared to V2R4.0 (increase ranges from [redacted]% to [redacted]%). The miscellaneous partition, as mentioned previously, comprises of RTS partition. RTS uses segment services for localized expression evaluation, which is causing higher CPU usage for the user services. The usage of segment services can be reduced, in order to reduce this overhead in V2R4.1.

- Total CPU usage percentage for User Service. See details in the spreadsheet attached below (TotalCPUTimeForUserService.xls):

TotalCPUTimeForUse
rService.xls

Conclusion

The percentage CPU usage for overall User service is less in V2R4.1 as compared to V2R4.0 (improvement ranges from [redacted]% to [redacted]%)..

- User Execution within the Parser partition of the PE. See details in the spreadsheet attached below (CPUTimeForExecutionInParserPartition.xls):

CPUTimeForExecutio
nInParserPartition.xls

Conclusion

The percentage CPU usage for execution within the Parser partition of PE is less in V2R4.1 as compared to V2R4.0 (improvement ranges from [redacted]% to [redacted]%).

- User Execution within the Dispatcher partition of the PE. See details in the spreadsheet attached below (CPUTimeForExecutionInDispatcherPartition.xls):

CPUTimeForExecutio
nInDispatcherPartitio

Conclusion

The percentage CPU usage for execution within the Dispatcher partition of PE is less in V2R4.1 as compared to V2R4.0 (improvement ranges from [redacted]% to [redacted]%).

- User Execution within the Miscellaneous partition of the PE. See details in the spreadsheet attached below (CPUTimeForExecutionInMiscPartition.xls):

CPUTimeForExecutio
nInMiscPartition.xls

Conclusion

The percentage CPU usage for execution within the Miscellaneous partitions of PE is more in V2R4.1 as compared to V2R4.0 (increase ranges from [redacted]% to [redacted]%). This indicates that the CPU usage in RTS is high in V2R4.1 as compared to V2R4.0. This is attributed to the localized expression evaluation in RTS.

- Total CPU usage percentage for User Execution. See details in the spreadsheet attached below (TotalCPUTimeForUserExecution.xls):

TotalCPUTimeForUse
rExecution.xls

Conclusion

The percentage CPU usage for overall User Execution is slightly more in V2R4.1 as compared to V2R4.0 (increase ranges from [redacted]% to [redacted]%). This is expected as the expression evaluation has shifted from Parser to RTS. Overall, the CPU usage percentage for User Execution does not vary drastically.

- Total CPU Time (For User Service and User Execution). See details in the spreadsheet attached below (TotalCPUTime.xls):

TotalCPUTime.xls

Conclusion

The percentage CPU usage is less in V2R4.1 as compared to V2R4.0 (improvement ranges from [redacted]% to [redacted]%), i.e. Stored procedure execution in V2R4.1 is less CPU intensive as compared to V2R4.0.

ResOneNode: This macro is used for data related to the Logical Device I/Os.

- Logical Device I/O. See details in the spreadsheet attached below (LDVIO Usage.xls):

"LDVIO Usage.xls"

Conclusion

In V2R4.1 there is significant reduction in the number of LDV I/Os when compared with the number of LDV I/Os in V2R4.0 (improvement ranges from [redacted]% to [redacted]%)..

Collection of Performance data using System Activity Reporter (SAR)

The following SAR command was used to log performance data using SAR:

```
sar -A -o file1  6 10000000
```

Here, the collection period is 6 seconds for 10000000 intervals. The value used for the number of intervals ensures that SAR data is collected even for the execution of stored procedure that takes maximum execution time. This is collected for each case, i.e., 1000, 2000 and 3000 iterations.

The following SAR command was used to get the I/O data related to Unix File System:

```
sar -d -f file1 -s start_time -e end_time
```

Start Time here refers to the start time of the stored procedure execution and end-time refers to the end time of the stored procedure execution. This is collected for each case, i.e., 1000, 2000 and 3000 iterations.

The following statistics were collected:

- Number of data transfers from and to device (changed from transfers/sec to #of transfers). See details in the spreadsheet attached below (NumberOfDataTransfers-SAR.xls):

NumberOfDataTransfers-SAR.xls

Conclusion

The number of disk reads and writes per second is less in V2R4.1 as compared to V2R4.0 (improvement is [redacted]%).

OVERALL CONCLUSIONS

- Stored procedure execution is faster in V2R4.1 when compared to V2R4.0

- Stored procedure execution in V2R4.1 is less CPU intensive as compared to V2R4.0.
- Number of Logical Disk I/Os is less in V2R4.1 as compared to V2R4.0.

Resource Usage Test Scripts

- Script to create the customized resource usage macros and the required tables.

```
CREATE SET TABLE RESUSER.respetab ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL
     (
      Node1 CHAR(6) CHARACTER SET UNICODE NOT CASESPECIFIC,
      Date1 DATE FORMAT 'YY/MM/DD',
      Time1 FLOAT,
      Vproc INTEGER,
      PEParsServ DECIMAL(12,5),
      PEDispServ DECIMAL(12,5),
      PESessServ DECIMAL(12,5),
      PEMiscUserServ DECIMAL(12,5),
      PEParsExec DECIMAL(12,5),
      PEDispExec DECIMAL(12,5),
      PESessExec DECIMAL(12,5),
      PEMiscUserExec DECIMAL(12,5),
      PETotalUserServ DECIMAL(12,5),
      PETotalUserExec DECIMAL(12,5),
      PEUserExecServ DECIMAL(12,5))
PRIMARY INDEX ( Node1 );
CREATE SET TABLE resuser.resnodetab ,NO FALLBACK ,
     NO BEFORE JOURNAL,
     NO AFTER JOURNAL
     (
      Node1 CHAR(6) CHARACTER SET KANJI1 NOT CASESPECIFIC,
      Date1 DATE FORMAT 'YY/MM/DD',
      Time1 FLOAT,
      CPUBsy DECIMAL(12,5),
      AvgIOs DECIMAL(12,5),
      TotPIOs DECIMAL(12,5),
      TotMIOs DECIMAL(12,5),
      TotNetPMIos DECIMAL(12,5))
PRIMARY INDEX ( Date1 );
REPLACE MACRO RESUSER.ResCPUByPEOneNode
   ( FromDate (DATE, DEFAULT DATE, FORMAT 'YYYY-MM-DD'),
     ToDate   (DATE, DEFAULT DATE, FORMAT 'YYYY-MM-DD'),
     FromTime (FLOAT, DEFAULT 0,      FORMAT'99:99:99'),
     ToTime   (FLOAT, DEFAULT 999999, FORMAT'99:99:99'),
     Node     (CHAR(6), DEFAULT '000-00') )
AS ( rollback 'Node parameter must be CHAR(6) in the format "999-99"'
               where index(:Node,'-') ^= 4;

     echo '.set heading "&DATE||CPU Usage by PE for Node &1||Page&PAGE"';
     echo '.set format on';
     echo '.set suppress on 2,3';
     echo '.set width 132';

     INSERT resuser.respetab
     SELECT  :Node,               /* This column omitted from report */

             TheDate (FORMAT 'yy/mm/dd', TITLE '// //    Date'),
             TheTime (FORMAT '99:99:99', TITLE '// //    Time'),
             Vproc   (FORMAT 'ZZZZ9',    TITLE '//Vproc//   Id'),

             /* Pct time PE Vprocs busy doing user service for ptn 14  */
             PEParsServ/Secs
             (FORMAT 'zz9.9%', TITLE '  Pars//  User// Serv%'),

             /* Pct time PE Vprocs busy doing user service for ptn 13  */
```

```
        PEDispServ/Secs
        (FORMAT 'zz9.9%', TITLE '  Disp//  User// Serv%'),

        /* Pct time PE Vprocs busy doing user service for ptn 12  */
        PESessServ/Secs
        (FORMAT 'zz9.9%', TITLE '   Ses//  User// Serv%'),

        /* Pct time PE Vprocs busy doing user service ptns 01-11,31 */
        PEMiscUserServ/Secs
        (FORMAT 'zz9.9%', TITLE '  Misc//  User// Serv%'),

        /* Pct time PE Vprocs busy doing user execution for ptn 14  */
        PEParsExec/Secs
        (FORMAT 'zz9.9%', TITLE '  Pars//  User// Exec%'),

        /* Pct time PE Vprocs busy doing user execution for ptn 13  */
        PEDispExec/Secs
        (FORMAT 'zz9.9%', TITLE '  Disp//  User// Exec%'),

        /* Pct time PE Vprocs busy doing user execution for ptn 12  */
        PESessExec/Secs
        (FORMAT 'zz9.9%', TITLE '   Ses//  User// Exec%'),

        /* Pct time PE Vprocs busy doing user exec ptns 01-11,31 */
        PEMiscUserExec/Secs
        (FORMAT 'zz9.9%', TITLE '  Misc//  User// Exec%'),

        /* Pct time PE Vprocs busy doing user service, all partitions */
        PETotalUserServ/Secs
        (FORMAT 'zz9.9%', TITLE ' Total//  User// Serv%'),

        /* Pct time PE Vprocs busy doing user exec, all partitions */
        PETotalUserExec/Secs
        (FORMAT 'zz9.9%', TITLE ' Total//  User// Exec%'),

        /* Pct time PE Vprocs busy doing user serv + exec, all ptns */
        (PETotalUserExec+PETotalUserServ)/Secs
        (FORMAT 'zz9.9%', TITLE ' Total//  Busy//     %')

FROM DBC.ResCPUUsageByPEView
WHERE ( ( ( TheDate = :FromDate AND TheTime >= :FromTime ) OR
          ( TheDate > :FromDate )
        ) AND
        ( ( TheDate = :ToDate   AND TheTime <= :ToTime   ) OR
          ( TheDate < :ToDate )
        ) AND

       NodeId = :Node
      );

SELECT
        /* Pct time PE Vprocs busy doing user service for ptn 14  */
        AVG(PEParsServ)
        (FORMAT 'zz9.9%', TITLE '  Pars//  User// Serv%'),

        /* Pct time PE Vprocs busy doing user service for ptn 13  */
        AVG(PEDispServ)
        (FORMAT 'zz9.9%', TITLE '  Disp//  User// Serv%'),

        /* Pct time PE Vprocs busy doing user service for ptn 12  */
        AVG(PESessServ)
        (FORMAT 'zz9.9%', TITLE '   Ses//  User// Serv%'),
            /* Pct time PE Vprocs busy doing user service ptns 01-11,31 */
            AVG(PEMiscUserServ)
            (FORMAT 'zz9.9%', TITLE '  Misc//  User// Serv%'),

            /* Pct time PE Vprocs busy doing user execution for ptn 14  */
            AVG(PEParsExec)
            (FORMAT 'zz9.9%', TITLE '  Pars//  User// Exec%'),

            /* Pct time PE Vprocs busy doing user execution for ptn 13  */
            AVG(PEDispExec)
            (FORMAT 'zz9.9%', TITLE '  Disp//  User// Exec%'),

            /* Pct time PE Vprocs busy doing user execution for ptn 12  */
            AVG(PESessExec)
            (FORMAT 'zz9.9%', TITLE '   Ses//  User// Exec%'),

            /* Pct time PE Vprocs busy doing user exec ptns 01-11,31 */
            AVG(PEMiscUserExec)
            (FORMAT 'zz9.9%', TITLE '  Misc//  User// Exec%'),

            /* Pct time PE Vprocs busy doing user service, all partitions */
            AVG(PETotalUserServ)
            (FORMAT 'zz9.9%', TITLE ' Total//  User// Serv%'),

            /* Pct time PE Vprocs busy doing user exec, all partitions */
            AVG(PETotalUserExec)
            (FORMAT 'zz9.9%', TITLE ' Total//  User// Exec%'),

            /* Pct time PE Vprocs busy doing user serv + exec, all ptns */
            AVG(PEUserExecServ)
            (FORMAT 'zz9.9%', TITLE ' Total//  Busy//     %')

    FROM RESUSER.respetab
    WHERE ( ( ( Date1 = :FromDate AND Time1 >= :FromTime ) OR
                ( Date1 > :FromDate )
            ) AND
            ( ( Date1 = :ToDate   AND Time1 <= :ToTime    ) OR
                ( Date1 < :ToDate )
            ) AND

           Node1 = :Node AND
              Vproc=16382
         );

    SELECT
            /* Pct time PE Vprocs busy doing user service for ptn 14  */
            AVG(PEParsServ)
            (FORMAT 'zz9.9%', TITLE '  Pars//  User// Serv%'),

            /* Pct time PE Vprocs busy doing user service for ptn 13  */
            AVG(PEDispServ)
            (FORMAT 'zz9.9%', TITLE '  Disp//  User// Serv%'),

            /* Pct time PE Vprocs busy doing user service for ptn 12  */
            AVG(PESessServ)
            (FORMAT 'zz9.9%', TITLE '   Ses//  User// Serv%'),

            /* Pct time PE Vprocs busy doing user service ptns 01-11,31 */
            AVG(PEMiscUserServ)
            (FORMAT 'zz9.9%', TITLE '  Misc//  User// Serv%'),

            /* Pct time PE Vprocs busy doing user execution for ptn 14  */
            AVG(PEParsExec)
```

```
          (FORMAT 'zz9.9%', TITLE '  Pars//  User// Exec%'),

          /* Pct time PE Vprocs busy doing user execution for ptn 13  */
          AVG(PEDispExec)
          (FORMAT 'zz9.9%', TITLE '  Disp//  User// Exec%'),

          /* Pct time PE Vprocs busy doing user execution for ptn 12  */
          AVG(PESessExec)
          (FORMAT 'zz9.9%', TITLE '   Ses//  User// Exec%'),

          /* Pct time PE Vprocs busy doing user exec ptns 01-11,31 */
          AVG(PEMiscUserExec)
          (FORMAT 'zz9.9%', TITLE '  Misc//  User// Exec%'),

          /* Pct time PE Vprocs busy doing user service, all partitions */
          AVG(PETotalUserServ)
          (FORMAT 'zz9.9%', TITLE ' Total//  User// Serv%'),

          /* Pct time PE Vprocs busy doing user exec, all partitions */
          AVG(PETotalUserExec)
          (FORMAT 'zz9.9%', TITLE ' Total//  User// Exec%'),

          /* Pct time PE Vprocs busy doing user serv + exec, all ptns */
          AVG(PEUserExecServ)
          (FORMAT 'zz9.9%', TITLE ' Total//  Busy//     %')

    FROM RESUSER.respetab
    WHERE ( ( ( Date1 = :FromDate AND Time1 >= :FromTime ) OR
              ( Date1 > :FromDate )
            ) AND
            ( ( Date1 = :ToDate   AND Time1 <= :ToTime   ) OR
              ( Date1 < :ToDate )
            ) AND

           Node1 = :Node AND
              Vproc=16383
         );

    echo '.set suppress off';
    echo '.set omit off';
    echo '.set format off';

  ) ;  /* end of MACRO ResCPUByPEOneNode */
;
REPLACE MACRO resuser.ResOneNode
  ( FromDate (DATE, DEFAULT DATE, FORMAT 'YYYY-MM-DD'),
    ToDate   (DATE, DEFAULT DATE, FORMAT 'YYYY-MM-DD'),
    FromTime (FLOAT, DEFAULT 0,      FORMAT'99:99:99'),
    ToTime   (FLOAT, DEFAULT 999999, FORMAT'99:99:99'),
    Node     (CHAR(6), DEFAULT '000-00')                    )

AS ( rollback 'Node parameter must be CHAR(6) in the format "999-99"'
              where index(:Node,'-') ^= 4;

    echo '.set heading "&DATE||General Resource Usage Summary||Page&PAGE"';
    echo '.set format on';
    echo '.set suppress 2';
    echo '.set width 132';

    INSERT resuser.resnodetab
    SELECT  /* First select list item omitted from report output */
          :Node,
        TheDate (FORMAT'yy/mm/dd', TITLE 'Date'),
        TheTime (FORMAT'99:99:99', TITLE 'Time'),

        /* Percent of time the CPUs were busy doing work */
        (CPUBusy)/Secs
        (FORMAT 'ZZ9', TITLE 'CPUBsy%'),

        /* Average number of logical device IOs per sec */
        (LogicalDeviceIO)/Secs
        (FORMAT 'ZZZ9', TITLE ' LdvIOs/Sec'),

        /* Tot point-to-point net reads/writes per sec */
        NetPtoPIO/Secs
        (FORMAT 'ZZZ9', TITLE 'TPtPIOs/Sec'),

        /* Tot multicast net reads/writes per sec */
        NetMultiIO/Secs
        (FORMAT 'ZZZ9', TITLE 'TMltIOs/Sec'),

        /* Tot point-to-point & multicast net reads/writes per sec */
        ((NetPtoPIO/Secs) + (NetMultiIO/Secs))
        (FORMAT 'ZZZZ9', TITLE 'TIOs')

FROM DBC.ResGeneralInfoView

WHERE ( ( ( TheDate = :FromDate AND TheTime >= :FromTime ) OR
          ( TheDate > :FromDate )
        ) AND
        ( ( TheDate = :ToDate   AND TheTime <= :ToTime   ) OR
          ( TheDate < :ToDate )
        ) AND
        NodeId = :Node
      );

/*
** Averages of CPU Busy time, Logical Device I/Os and
** Total BYNET Traffic for the total CALL execution time.
*/

SELECT (AVG(CPUBsy))
       (FORMAT 'ZZZ9.99', TITLE 'CPUBsy'),

       (AVG(AvgIOs))
       (FORMAT 'ZZZ9.99', TITLE 'LdvIOs'),

       (AVG(TotPIOs))
       (FORMAT 'ZZZ9.99', TITLE 'TPtPIOs'),

       (AVG(TotMIOs))
       (FORMAT 'ZZZ9.99', TITLE 'TMltIOs'),

       (AVG(TotNetPMIos))
       (FORMAT 'ZZZZ9.99', TITLE 'TIOs')

 FROM resuser.resnodetab

 WHERE ( ( ( Date1 = :FromDate AND Time1 >= :FromTime ) OR
           ( Date1 > :FromDate )
         ) AND
         ( ( Date1 = :ToDate   AND Time1 <= :ToTime   ) OR
           ( Date1 < :ToDate )
         ) AND
        Node1 = :Node
```

```
          );
     echo '.set suppress off';
     echo '.set omit off';
     echo '.set format off';
   ) ;  /* end of MACRO ResOneNode */
;
```

- Test Script used to create the stored procedures:

```
.logon dbc,dbc
 sel * from dbcinfo;
 delete user resuser;
 drop user resuser;
 create user resuser as password=resuser, perm=10e6;
 grant all on resuser to resuser;
 grant all on dbc to resuser with grant option;
.logon resuser, resuser
.compile file penest11.spl
 show procedure penest11;
.compile file penest1.spl
 show procedure penest1;

.compile file penest22.spl
 show procedure penest22;
.compile file penest2.spl
 show procedure penest2;

.compile file penest33.spl
 show procedure penest33;
.compile file penest3.spl
 show procedure penest3;

.logoff
.quit;
```

- Test Script used to execute the stored procedures and collect the performance data:

```
.logon resuser, resuser
 sel time;
 call penest1(1000);
 sel time;
 call penest2(2000);
 sel time;
 call penest3(3000);
 sel time;
.quit;
```

System Configurations

The following systems are used for performance testing.

SMP 4400 (pennar)

Number of Nodes	1
Number of Processors per Node	4
Speed of the each Processor	550 MHz, P III
RAM per Node	2GB
Number of AMPS	8
Number of Clusters	1
Number of VPROC's(AMP) per node	8
Number of PDISKs per VPROC	1
Hard Disk Capacity	(2GBx8=16GB)

MPP 5250 (agni)

Number of Nodes	4
Number of Processors per Node	4
Speed of the each Processor	500 MHz, P III
RAM per Node	2GB
Number of AMPS	16
Number of Clusters	4
Number of VPROC's(AMP) per node	4
Number of PDISKs per VPROC	1
Hard Disk Capacity	(8GBx4=32GB per node, 2disks for system, 2 disks for database on all 4 nodes)

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** ______________________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.